Panel Pointcloud Registration

Wondong Kang

February 2025

1. Abstract

This paper proposes a structured pipeline for robust point cloud registration utilizing notch-based alignment techniques. The method consists of a series of processing steps, including transformation, downsampling, notch detection, and refinement, ensuring accurate and interpretable alignment.

The approach begins with preprocessing the reference point cloud by applying homogeneous coordinate transformations, voxel-based downsampling, and segmentation techniques to identify notch features. Real-world point clouds undergo statistical noise filtering and clustering to enhance data quality and ensure comparability with the reference dataset. Notch detection is performed through spatial segmentation along multiple axes, leveraging clustering algorithms to identify and highlight key alignment features.

To estimate the initial transformation matrix, Principal Component Analysis (PCA) is employed to align detected notch regions between the reference and real-world point clouds. Postprocessing involves further refinement through targeted rotations and translations based on cluster alignment, ensuring improved precision across different perspectives, including Top, Bottom, and Side views.

Experimental results validate the effectiveness of the proposed pipeline, demonstrating successful alignment of real-world point clouds with reference datasets while preserving notch feature integrity.

2. Method

2-1. Reference Pointclouds Processing

Point Cloud Transformation and Color Assignment

Process Description: This stage involves converting 2D image data into 3D point cloud data. The transformation applies homogeneous coordinate transformations to each pixel's 3D coordinates, converting them into a spatial coordinate system.

Mathematical Background: Each point (x, y, z) undergoes a homogeneous coordinate transformation to calculate its new position (x', y', z'):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where T is a 4x4 transformation matrix that dictates the point cloud's rotation, translation, and scaling.

Downsampling

Process Description: The downsampling step reduces the volume of data in the point cloud using a 'VoxelGrid' filter. This process simplifies each voxel's points into a single representative point to enhance processing speed and decrease noise.

Mathematical Background: The 'VoxelGrid' calculates the centroid of points within each voxel to establish a representative point:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i, \quad \bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i$$

where N is the number of points in each voxel.

Reference Notch Detection

In this section, I describe a robust method for detecting notches in point cloud data. The process adapts based on the input image file type and can be summarized as follows:

- **Case 1: MASTER_SIDE.tiff**
  The point cloud is first sorted by the x-coordinate and split into two clusters at the largest x-gap. The cluster with the lower average z-value is chosen as the target cluster. A search radius is computed based on the spatial extent of the target cluster, and the cluster is segmented along the y-axis. For each segment, the average z-value is determined, and the top three segments (by highest average z) are selected. Points within these segments (from both target and non-target clusters) are then colored (red, green, and blue) to highlight the detected notches.

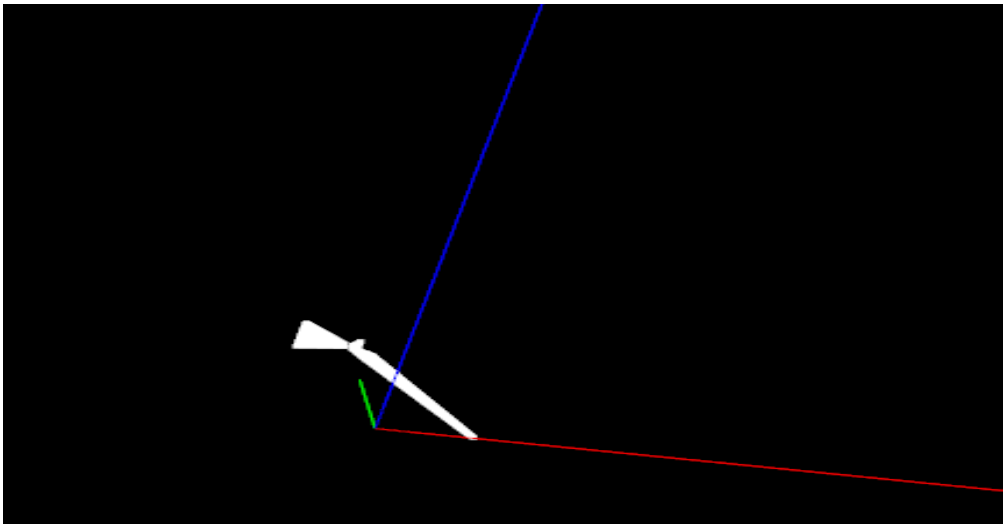


Figure 1: A partial view of the `MASTER_SIDE.tiff` dataset showing the notch detection process.

Figure 1 illustrates a segment of the `MASTER_SIDE.tiff` point cloud. Due to the characteristics of this dataset, the cloud can be clearly split into two clusters along the x-axis. Among these clusters, the one with the smaller average z-value is chosen as it typically contains the notch region. Next, by further clustering based on z-values, the subcluster with higher z is selected, effectively isolating the notch for visualization and subsequent analysis.

- **Case 2: General Case**
  For other file types, the entire point cloud is considered. A search radius is defined based on the x-axis extent. The cloud is segmented along the y-axis, and for each region, the number of points within the search radius is calculated. After removing the boundary regions, the three regions with the lowest point counts are identified and colored (red, green, and blue) to mark the notch areas.

Figure 2: A partial view of the dataset illustrating the notch detection process by counting points within a circular region along the y-axis.

Figure 2 shows the approach of scanning along the y-axis with a circular region. At each step, a circle is centered at:

$$(x_{\text{center}},\, y,\, z_{\text{center}})$$

with a predefined radius. All points lying inside this circle are counted. Regions where the point count is notably lower are identified as potential notches, since a notch typically appears as a local decrease in point density when viewed from this perspective.

The following detailed explanation breaks down each step mathematically.

Detailed Explanation

1. Case 1: MASTER_SIDE.tiff Processing
1.1 Cloud Splitting and Clustering:

1. **Sorting:**
   Given a point cloud with each point $p_i = (x_i, y_i, z_i)$, sort the points in ascending order of the x-coordinate:
   $$x_i < x_j \quad \text{for } i < j.$$
2. **Cluster Division:**
   Compute the gap between consecutive points:
   $$\Delta x_i = x_{i+1} - x_i, \quad i = 1, 2, \ldots, N-1.$$
   Identify the maximum gap $\Delta x_{\max}$ at index i^* and split the cloud into:
   $$\text{Cluster1: } \{p_1, \ldots, p_{i^*}\}, \quad \text{Cluster2: } \{p_{i^*+1}, \ldots, p_N\}.$$

3. **Target Cluster Selection:**
   Compute the average z-value for each cluster:

$$\bar{z}_1 = \frac{1}{n_1} \sum_{p \in \text{Cluster1}} p.z, \quad \bar{z}_2 = \frac{1}{n_2} \sum_{p \in \text{Cluster2}} p.z.$$

   The cluster with the lower $\bar{z}$ is selected as the *target cluster*.

1.2 Search Area Definition and y-axis Segmentation:

1. **Boundary Determination:**
   Determine the minimum and maximum coordinates of the target cluster:

$$\text{minPt} = (x_{\min}, y_{\min}, z_{\min}), \quad \text{maxPt} = (x_{\max}, y_{\max}, z_{\max}).$$

2. **Radius and Center Calculation:**
   Calculate the search radius based on the extent in the x and z dimensions:

$$r = 2\sqrt{(x_{\max} - x_{\min})^2 + (z_{\max} - z_{\min})^2}.$$

   The center coordinates are:

$$\text{center}_x = \frac{x_{\min} + x_{\max}}{2}, \quad \text{center}_z = \frac{z_{\min} + z_{\max}}{2}.$$

3. **y-axis Segmentation:**
   For each y value in the range $[y_{\min}, y_{\max}]$ with increments of r, perform a KD-tree radius search around the point:

$$(\text{center}_x,\ y,\ \text{center}_z)$$

   to collect all points within radius r. For each segment, compute the average z-value:

$$\bar{z}_{\text{segment}} = \frac{1}{N} \sum_{p \in \text{segment}} p.z.$$

4. **Segment Selection and Coloring:**
   Sort the segments in descending order by $\bar{z}_{\text{segment}}$ and select the top three. Assign each segment a distinct color (red, green, blue). For each segment, color all points (from both target and non-target clusters) satisfying:

$$|p.y - y_{\text{center}}| \leq r.$$

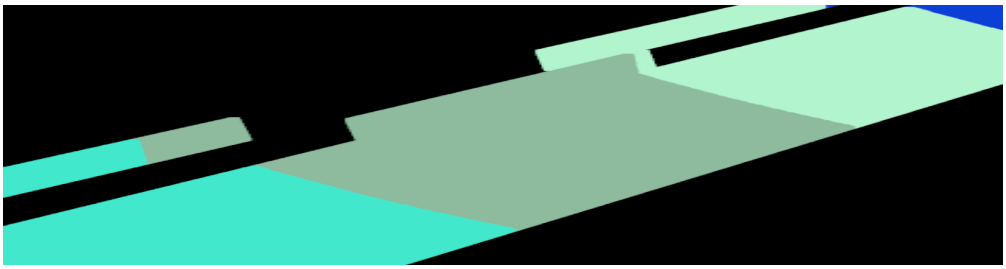


Figure 3: A figure illustrating the notch derived from the aforementioned formulas. This visualization highlights the regions identified as notches after applying the x- and z-axis segmentation and clustering approach described earlier.

2. Case 2: General Processing

1. **Boundary and Radius Determination:**
   For the entire point cloud, determine:

$$\text{minPt} = (x_{\min}, y_{\min}, z_{\min}), \quad \text{maxPt} = (x_{\max}, y_{\max}, z_{\max}).$$

   The search radius is defined as:

$$r = 7\frac{x_{\max} - x_{\min}}{2},$$

   and the center is given by:

$$\text{center}_x = x_{\min} + r, \quad \text{center}_z = \frac{z_{\min} + z_{\max}}{2}.$$

2. **y-axis Region Analysis:**
   Segment the cloud along the y-axis with increments of r. For each region centered at:

$$(\text{center}_x,\ y,\ \text{center}_z),$$

   count the number of points within radius r:

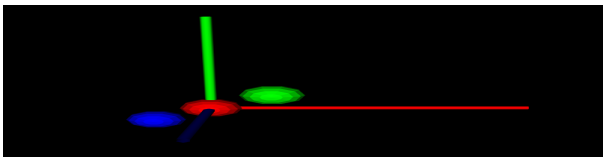
$$\text{count}(y) = \text{number of points within } r.$$

3. **Region Selection and Coloring:**
   Remove the boundary regions, sort the remaining regions by count(y) in ascending order, and select the three regions with the lowest counts. For each selected region, color points that satisfy:

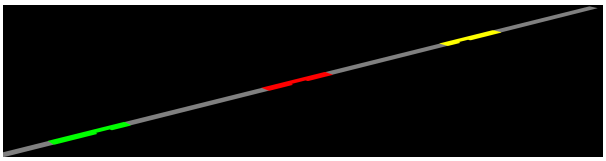
$$\text{center}_x - r \leq p.x \leq \text{center}_x + r, \quad y_{\text{center}} - r \leq p.y \leq y_{\text{center}} + r.$$

   Each region is assigned one of the colors red, green, or blue.

This method effectively identifies and visualizes notch areas in point clouds by leveraging spatial segmentation and clustering techniques, making it a valid and robust approach for notch detection.



(a) Visualization of the dynamic radius search along the Y-axis.



(b) Result showing transformation, downsampling, and notch detection.

Figure 4: Illustrations of notch detection and feature isolation techniques used in point cloud processing. The figures show two stages: the detection methodology and the resultant visualizations.

Figure 5: Visualization of notch detection results from different perspectives (top, bottom, and side) using the reference notch detection algorithm. Each image highlights detected notches in distinct colors to enhance clarity.

2-2. Real Pointclouds Processing

In practical applications, obtaining pristine point cloud data comparable to reference point clouds is challenging due to occlusions and various noise factors. To address these challenges, it is essential to employ statistical noise removal and clustering techniques. These methods enhance the quality of the data and are crucial for ensuring accuracy in subsequent processing steps.

Statistical Noise Removal and Clustering Techniques

Processing real-world point cloud data requires meticulous noise filtering and data segmentation to ensure the data quality is on par with that of reference datasets. The following steps outline the methodologies applied:

1. **Filtering Based on X-Axis Width:** Clusters with an X-axis width less than 30% of the total point cloud width are considered noise and are filtered out. This approach is based on the assumption that significant features of the point cloud will span a broader X-axis range.

$$\text{Threshold Width} = 0.3 \times (\text{maxPt.x} - \text{minPt.x})$$

2. **Detection of Breaks Along the Y-Axis:** The point cloud is analyzed for significant gaps along the Y-axis. Points are sorted by their Y-values, and large differences between successive points indicate potential breaks or gaps.

$$\text{Break Detected If } \Delta y > \text{threshold}$$

3. **Filtering Top Two Y-Axis Segments:** Only the segments of the point cloud that represent the longest contiguous Y-axis extents are retained. This method ensures focus on the most significant parts of the point cloud.

$$\text{Segment Length} = y_{\text{end}} - y_{\text{start}}$$

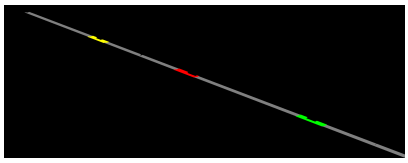
4. **Removal of Small Components:** Using Euclidean clustering, small clusters that do not meet specific height and area criteria are removed. This refinement step focuses on the significant components of the point cloud.

$$\begin{aligned} \text{Height} &= \text{maxPt.y} - \text{minPt.y} \\ \text{Area} &= (\text{maxPt.x} - \text{minPt.x}) \times (\text{maxPt.z} - \text{minPt.z}) \end{aligned}$$

These processes not only facilitate the visualization and analysis of structural features but also play a pivotal role in applications such as industrial automation, where accurate data is crucial for operational efficiency and safety.

(a) This image displays the raw real data point cloud showing noise and occlusions, illustrating discontinuities and irregular point distributions.

(b) After applying statistical and clustering analysis, noise elements are removed, retaining the shape of the panel as much as possible in the point cloud.



(c) The result of applying the same notch detection process used on the reference point cloud to the noise-reduced real data point cloud, demonstrating the detection of panel notches.

Figure 6: Illustrations of the real data processing at various stages: initial raw data, noise reduction through statistical and clustering analysis, and final notch detection. These steps illustrate the transformation of the data from raw to processed, maintaining the integrity and structural details of the panel.

Real Notch Detection

The Real Notch Detection process follows a method very similar to that of the Reference Notch Detection. However, in the case of the `master_side.tiff` dataset, significant data degradation occurs due to occlusion or noise. To address this, the method is adjusted by combining Z-axis sorting with Y-axis segmentation.

First, the point cloud is sorted by the z-coordinate:

$$p_i = (x_i,\, y_i,\, z_i) \quad \text{with} \quad z_1 \leq z_2 \leq \cdots \leq z_N.$$

The largest gap in the z-direction is then identified:

$$\Delta z_i = z_{i+1} - z_i, \quad \Delta z_{\max} = \max_{1 \leq i < N} (\Delta z_i),$$

which is used to split the cloud into two clusters. The average z-value for each cluster is computed as follows:

$$\bar{z}_1 = \frac{1}{n_1} \sum_{p \in \text{Cluster1}} p.z, \quad \bar{z}_2 = \frac{1}{n_2} \sum_{p \in \text{Cluster2}} p.z.$$

The cluster with the higher average z (i.e., when $\bar{z}_1 > \bar{z}_2$) is selected as the target cluster.

Next, the target cluster is segmented along the y-axis. For each y value within the target cluster, a circular search region is defined with center

$$(x_{\text{center}},\, y,\, z_{\text{center}})$$

and a predefined radius R. The number of points within each circular region is then counted:

$$\text{count}(y) = \#\{\, p \mid \sqrt{(p.x - x_{\text{center}})^2 + (p.z - z_{\text{center}})^2} \leq R \,\}.$$

Regions with notably lower point counts are identified as notch candidates.

By combining Z-axis sorting with Y-axis segmentation, this method effectively mitigates the impact of occlusion and noise present in the `master_side.tiff` dataset, leading to a robust detection of the notch.

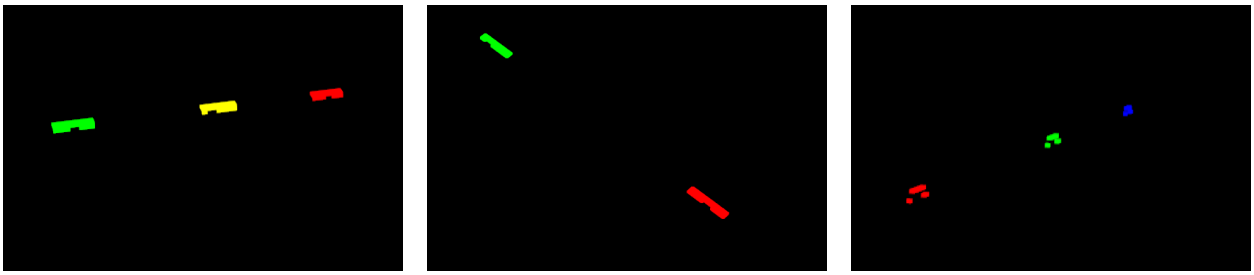


Figure 7: Visualization of notch detection results from different perspectives (top, bottom, and side) using the Real Notch Detection algorithm. This approach refines detection by addressing occlusions and noise using Z-axis sorting and Y-axis segmentation.

2-3. Initial Alignment Using Real and Reference Notches

In this section, I compute the initial transformation matrix **T** that aligns the **Real Notch** and **Reference Notch** point clouds. The method utilizes Principal Component Analysis (PCA) for feature extraction and transformation estimation.

Normal Vector Calculation

Normal vectors provide geometric structure information and are computed for both the source (Real Notch) and target (Reference Notch) point clouds.

1. **Nearest Neighbor Search for Normal Estimation** For each point $\mathbf{p}_i = (x_i, y_i, z_i)$, I determine the k nearest neighbors and compute the covariance matrix:

$$\mathbf{C}_i = \sum_{j \in \mathcal{N}_k(i)} (\mathbf{p}_j - \bar{\mathbf{p}})(\mathbf{p}_j - \bar{\mathbf{p}})^T$$

where $\mathcal{N}_k(i)$ represents the set of k nearest neighbors and $\bar{\mathbf{p}}$ is the mean position.

2. **Eigenvalue Decomposition for Normal Vector Computation** The eigenvector corresponding to the smallest eigenvalue of $\mathbf{C}_i$ is selected as the normal vector:

$$\mathbf{C}_i \mathbf{n}_i = \lambda_{\min} \mathbf{n}_i$$

Principal Component Analysis (PCA) for Alignment

To align the Real Notch and Reference Notch point clouds, I use PCA to determine their principal axes.

1. **Compute Centroids** I calculate the centroids of both point clouds:

$$\mathbf{c}_S = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i^{(S)}, \quad \mathbf{c}_T = \frac{1}{M} \sum_{i=1}^{M} \mathbf{p}_i^{(T)}$$

where $\mathbf{p}_i^{(S)}$ and $\mathbf{p}_i^{(T)}$ denote points from the source (Real Notch) and target (Reference Notch), respectively.

2. **Compute Covariance Matrices** The normalized covariance matrices for the source and target clouds are:

$$\mathbf{\Sigma}_S = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{p}_i^{(S)} - \mathbf{c}_S)(\mathbf{p}_i^{(S)} - \mathbf{c}_S)^T$$

$$\mathbf{\Sigma}_T = \frac{1}{M} \sum_{i=1}^{M} (\mathbf{p}_i^{(T)} - \mathbf{c}_T)(\mathbf{p}_i^{(T)} - \mathbf{c}_T)^T$$

3. **Compute Principal Axes** By performing eigenvalue decomposition:

$$\mathbf{\Sigma}_S \mathbf{v}_S = \lambda_S \mathbf{v}_S, \quad \mathbf{\Sigma}_T \mathbf{v}_T = \lambda_T \mathbf{v}_T$$

the eigenvectors $\mathbf{v}_S$ and $\mathbf{v}_T$ define the principal axes.

4. **Compute Rotation Matrix** The initial rotation matrix $\mathbf{R}$ is determined by aligning the eigenvectors:

$$\mathbf{R} = \mathbf{V}_T \mathbf{V}_S^T$$

5. **Compute Translation Vector** The translation component is obtained by:

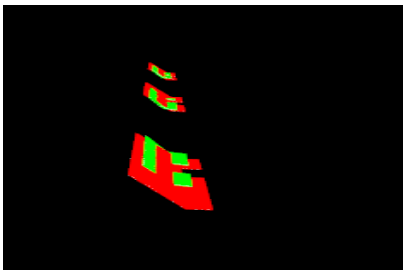
$$\mathbf{t} = \mathbf{c}_T - \mathbf{R}\mathbf{c}_S$$

Constructing the Transformation Matrix

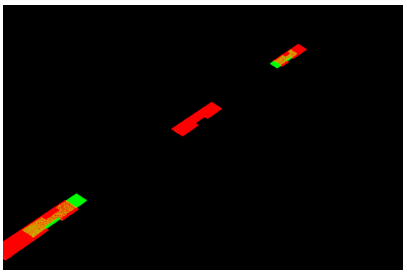
The final initial transformation matrix $\mathbf{T}$ is given by:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

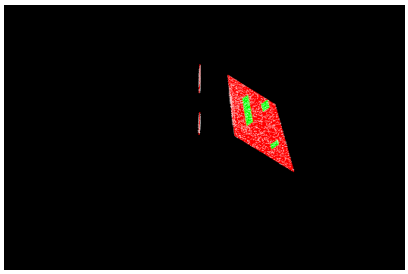
This transformation is applied to the source (Real Notch) cloud to bring it into alignment with the target (Reference Notch) cloud. The results are visualized with the transformed source in green and the target in red.



(a) Top view of the initial alignment using the computed transformation matrix.



(b) Bottom view of the initial alignment, showing the alignment between Real Notch and Reference Notch.



(c) Side view of the initial alignment result after applying the estimated transformation matrix.

Figure 8: Visualization of the initial alignment process where the **Real Notch** was aligned to the **Reference Notch** using the computed transformation matrix. The transformation matrix was derived through the PCA-based alignment method described earlier.

2-4. Postprocessing and Final Transformation Matrix

This section presents the final refinement steps for aligning the Real Notch with the Reference Notch using a sequence of transformations. The process varies for the **Side**, **Top**, and **Bottom** alignments, each applying specific adjustments to ensure precise registration.

Side Alignment Process

In this process, the initial transformation matrix T_1 is computed using the Reference Notch and Real Notch. Subsequently, a second alignment is performed to obtain the final transformation matrix T_2. The key steps are as follows:

1. **First Alignment using Initial Transformation**

   The initial transformation matrix T_1 is applied to transform the Real Notch:

   $$P_{\text{rotated}} = T_1 P_{\text{real}}$$

   After this transformation, the rotated Real Notch P_{rotated} is clustered based on the Z-axis.

2. **Clustering along the Z-Axis**

   The transformed Real Notch P_{rotated} is sorted along the Z-axis and split into two clusters:

   (a) **Sorting by Z-values**

   The point cloud is sorted in ascending order of Z-coordinates:

   $$z_1 \leq z_2 \leq \cdots \leq z_N$$

   (b) **Finding the Largest Z-Gap**

   The largest difference between consecutive Z-values is determined:

   $$\Delta z_i = z_{i+1} - z_i, \quad i = 1, 2, \ldots, N-1$$

   $$\Delta z_{\max} = \max(\Delta z_i)$$

   The clustering is performed at the position of the maximum gap:

   $$P_{\text{lowZ}} = \{p_i \in P_{\text{rotated}} \mid i < i^*\}, \quad P_{\text{highZ}} = \{p_i \in P_{\text{rotated}} \mid i \geq i^*\}$$

   The cluster with lower Z-values (P_{lowZ}) is colored yellow, and the one with higher Z-values (P_{highZ}) is colored green.

3. **Computing the Rotation Transformation for Cluster Alignment**

After Z-based clustering, P_{highZ} is aligned with its corresponding Reference Notch cluster.

(a) **Computing Normal Vectors Using PCA**

The normal vectors of each cluster are computed using Principal Component Analysis (PCA):

$$C_{\text{highZ}} = \sum_{i=1}^{N} (p_i - c_{\text{highZ}})(p_i - c_{\text{highZ}})^T$$

$$C_{\text{ref}} = \sum_{i=1}^{M} (p_i - c_{\text{ref}})(p_i - c_{\text{ref}})^T$$

The normal vectors correspond to the eigenvectors associated with the smallest eigenvalues:

$$C_{\text{highZ}} n_{\text{highZ}} = \lambda_{\min} n_{\text{highZ}}, \quad C_{\text{ref}} n_{\text{ref}} = \lambda_{\min} n_{\text{ref}}$$

(b) **Computing Rotation Axis and Angle**

The rotation axis and angle between the two normal vectors are calculated as:

$$r = n_{\text{highZ}} \times n_{\text{ref}}, \quad r = \frac{r}{||r||}$$

$$\theta = \cos^{-1}(n_{\text{highZ}} \cdot n_{\text{ref}})$$

(c) **Constructing the Rotation Matrix**

Using the computed rotation axis and angle, the rotation matrix is formed as:

$$R = \exp(\theta [r]_\times)$$

where $[r]_\times$ is the skew-symmetric matrix:

$$[r]_\times = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

The second transformation matrix T_2 is then constructed as:

$$T_2 = \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix}$$

Applying T_2, the second transformation is performed:

$$P_{\text{aligned}} = T_2 P_{\text{rotated}}$$

4. **Clustering along the X-Axis**

After the second transformation, additional clustering is performed along the X-axis.

(a) **Sorting by X-values**

The point cloud is sorted in ascending order of X-coordinates:

$$x_1 \leq x_2 \leq \cdots \leq x_N$$

(b) **Finding the Largest X-Gap**

The largest gap along X is found:

$$\Delta x_i = x_{i+1} - x_i, \quad \Delta x_{\max} = \max(\Delta x_i)$$

The clustering is performed at the location of $\Delta x_{\max}$:

$$P_{\text{leftX}} = \{p_i \mid i < i^*\}, \quad P_{\text{rightX}} = \{p_i \mid i \geq i^*\}$$

The left cluster (P_{leftX}) is colored blue, and the right cluster (P_{rightX}) is colored red.

5. **Computing the Final Translation Transformation**

   Finally, a translation transformation is applied based on the centroids of the aligned clusters.

   (a) **Computing Centroids**

   The centroids of the clusters are computed as:

$$c_{\text{real}} = \frac{1}{N}\sum_{i=1}^{N} p_i, \quad c_{\text{ref}} = \frac{1}{M}\sum_{i=1}^{M} p_i$$

   (b) **Computing Translation Vector**

   The translation vector is given by:

$$t = c_{\text{ref}} - c_{\text{real}}$$

   (c) **Constructing the Final Transformation Matrix**

   The final transformation matrix is:

$$T_{\text{final}} = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix}$$

   Applying T_{final}, the final alignment is completed:

$$P_{\text{final}} = T_{\text{final}} P_{\text{aligned}}$$

Top Alignment Process

The top alignment process consists of applying the initial transformation matrix T_1, followed by clustering based on the $X - Z$ plane, applying a second rotation transformation matrix T_2, and finally performing translation alignment using the transformation matrix T_3. The final transformation matrix T_{final} is then obtained.

The key steps are as follows:

1. **Applying the Initial Transformation Matrix**

   The initial transformation matrix T_1 is applied to align the Real Notch with the Reference Notch:

$$P_{\text{rotated}} = T_1 P_{\text{real}}$$

2. **Clustering along the X-Axis**

   The Reference Notch and the Real Notch are split into two clusters based on the X-axis:

$$P_{X1}, P_{X2} = \text{splitReferenceNotchesByX}(P_{\text{ref}})$$

$$P_{RXZ1}, P_{RXZ2} = \text{splitReferenceNotchesByXZ}(P_{\text{rotated}})$$

   (a) **Sorting by X-Values**

   The point cloud is sorted in ascending order of X-coordinates:

$$x_1 \leq x_2 \leq \cdots \leq x_N$$

   (b) **Finding the Largest X-Gap**

   The largest difference between consecutive X-values is determined:

$$\Delta x_i = x_{i+1} - x_i, \quad i = 1, 2, \ldots, N-1$$

$$\Delta x_{\max} = \max(\Delta x_i)$$

   The clustering is performed at the location where $\Delta x_{\max}$ occurs:

$$P_{X1} = \{p_i \mid i < i^*\}, \quad P_{X2} = \{p_i \mid i \geq i^*\}$$

   Here, P_{X1} is colored blue, and P_{X2} is colored red.

3. **Clustering along the X-Z Plane**

   The largest Euclidean distance between consecutive points in the X-Z plane is identified to split the clusters.

   (a) **Sorting by X-Z Values**

   The points are sorted primarily by X-coordinates and secondarily by Z-coordinates.

   (b) **Finding the Largest X-Z Distance**

   The Euclidean distance in the X-Z plane is computed:

$$\Delta XZ_i = \sqrt{(x_{i+1} - x_i)^2 + (z_{i+1} - z_i)^2}$$

$$\Delta XZ_{\max} = \max(\Delta XZ_i)$$

   The clustering is performed at the position of the maximum distance:

$$P_{RXZ1} = \{p_i \mid i < i^*\}, \quad P_{RXZ2} = \{p_i \mid i \geq i^*\}$$

4. **Computing the Rotation Transformation for Cluster Alignment**

   After X-Z based clustering, P_{RXZ2} is aligned with its corresponding Reference Notch cluster.

   (a) **Computing Normal Vectors Using PCA**

   The normal vectors of each cluster are computed using Principal Component Analysis (PCA):

$$C_{RXZ2} = \sum_{i=1}^{N} (p_i - c_{RXZ2})(p_i - c_{RXZ2})^T$$

$$C_{X2} = \sum_{i=1}^{M} (p_i - c_{X2})(p_i - c_{X2})^T$$

   The normal vectors correspond to the eigenvectors associated with the smallest eigenvalues:

$$C_{RXZ2} n_{RXZ2} = \lambda_{\min} n_{RXZ2}, \quad C_{X2} n_{X2} = \lambda_{\min} n_{X2}$$

   (b) **Computing Rotation Axis and Angle**

   The rotation axis and angle between the two normal vectors are calculated as:

$$r = n_{RXZ2} \times n_{X2}, \quad r = \frac{r}{||r||}$$

$$\theta = \cos^{-1}(n_{RXZ2} \cdot n_{X2})$$

   (c) **Constructing the Rotation Matrix**

   Using the computed rotation axis and angle, the rotation matrix is formed as:

$$R = \exp(\theta [r]_\times)$$

   where $[r]_\times$ is the skew-symmetric matrix:

$$[r]_\times = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

   The second transformation matrix T_2 is then constructed as:

$$T_2 = \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix}$$

   Applying T_2, the second transformation is performed:

$$P_{\text{aligned}} = T_2 P_{\text{rotated}}$$

5. **Computing the Final Translation Transformation**

   Finally, a translation transformation is applied based on the centroids of the aligned clusters.

   (a) **Computing Minimum X-Value for Filtering**

   The minimum X-value of R_{XZ2} is determined and scaled:

$$X_{\min,R} = X_{\min}(R_{XZ2}) \times 0.9$$

   (b) **Computing Centroids**

   The centroids of the clusters are computed as:

$$c_{\text{real}} = \frac{1}{N}\sum_{i=1}^{N} p_i, \quad c_{\text{ref}} = \frac{1}{M}\sum_{i=1}^{M} p_i$$

   (c) **Computing Translation Vector**

   The translation vector is given by:

$$t = c_{X2,\text{filtered}} - c_{RXZ2}$$

   (d) **Constructing the Final Transformation Matrix**

   The final transformation matrix is:

$$T_{\text{final}} = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix}$$

   Applying T_{final}, the final alignment is completed:

$$P_{\text{final}} = T_{\text{final}} P_{\text{aligned}}$$

Bottom Alignment Process

The bottom alignment process consists of applying the initial transformation matrix T_1, followed by a fixed reflection transformation T_2, and finally performing translation alignment using the transformation matrix T_3. The final transformation matrix T_{final} is then obtained.

The key steps are as follows:

1. **Applying the Initial Transformation Matrix**

   The initial transformation matrix T_1 is applied to align the Real Notch with the Reference Notch:

$$P_{\text{rotated}} = T_1 P_{\text{real}}$$

2. **Performing Reflection Along the X-Axis**

   In the bottom alignment process, the Real Notch is reflected along the X-axis to match the orientation of the Reference Notch. The second transformation matrix T_2 is defined as:

$$T_2 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   Applying this transformation inverts the X-coordinates:

$$P_{\text{second rotated}} = T_2 P_{\text{rotated}}$$

3. **Computing the Translation Transformation for Center Alignment**

   After the reflection transformation, a translation transformation is applied to align the centroid of the Real Notch with that of the Reference Notch.

(a) **Filtering Based on Minimum and Maximum Y-Values**
The minimum and maximum Y-values of the Real Notch cluster R_{Z2} are computed:

$$Y_{\min,R} = Y_{\min}(R_{Z2}) \times 0.901$$

$$Y_{\max,R} = Y_{\max}(R_{Z2}) \times 1.017$$

The points in the Reference Notch cluster C_{Z2} that satisfy the range constraint are selected:

$$Y_{\min,R} \leq Y \leq Y_{\max,R}$$

A new filtered cluster $C_{Z2,\text{filtered}}$ is then created.

(b) **Computing Centroids for Translation Alignment**
The centroids of R_{Z2} and $C_{Z2,\text{filtered}}$ are calculated as follows:

$$c_R = \frac{1}{|R_{Z2}|} \sum_{p \in R_{Z2}} p$$

$$c_C = \frac{1}{|C_{Z2,\text{filtered}}|} \sum_{p \in C_{Z2,\text{filtered}}} p$$

The translation vector is then computed as:

$$t = c_C - c_R$$

(c) **Constructing the Translation Transformation Matrix**
The translation transformation matrix T_3 is constructed using t:

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, applying T_3, the Real Notch is aligned with the Reference Notch:

$$P_{\text{final}} = T_3 P_{\text{second rotated}}$$

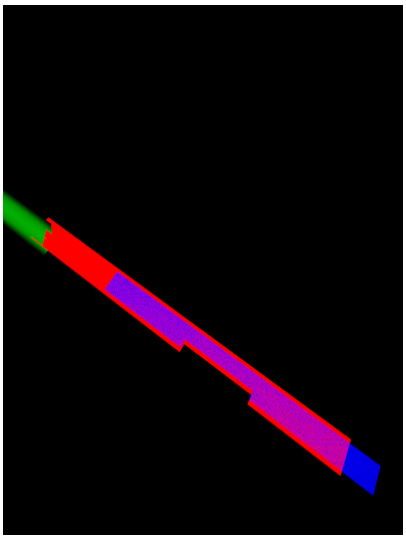
4. **Computing the Final Transformation Matrix**
The final transformation matrix T_{final} is obtained by combining all three transformations:

$$T_{\text{final}} = T_3 T_2 T_1$$

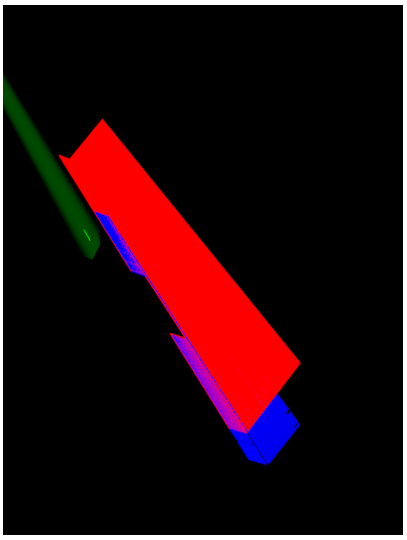
Applying T_{final}, the fully aligned Real Notch point cloud is obtained:

$$P_{\text{final}} = T_{\text{final}} P_{\text{real}}$$

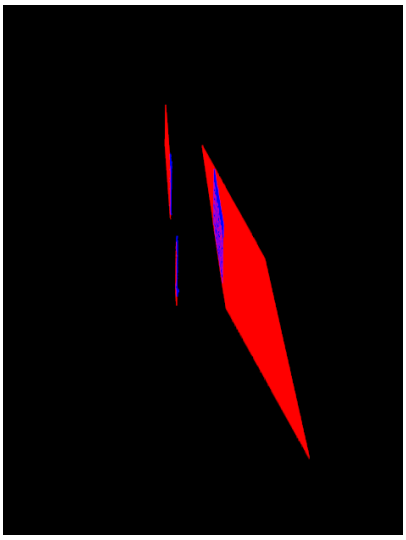
The transformed P_{final} is then compared with the Reference Notch to verify the final alignment.



(a) Final registration result for the **Top** case, aligning the real point cloud (blue) with the reference data (red). Notch regions are visualized for clarity.



(b) Final registration result for the **Bottom** case, where the real point cloud (blue) is aligned with the reference data (red). Notch details are highlighted.



(c) Final registration result for the **Side** case, aligning the real data (blue) with the reference data (red). The notch visualization ensures precise alignment.

Figure 9: Final point cloud registration results for the **Top**, **Bottom**, and **Side** cases. The real data (blue) has been aligned with the reference data (red), ensuring accurate notch alignment.

3. Conclusion

This paper presents a structured pipeline for point cloud alignment based on notch detection and transformation estimation. The method effectively registers real-world point clouds with reference datasets by systematically processing the data through transformation, downsampling, notch detection, and iterative refinement.

Our approach ensures precise alignment through:

- **Preprocessing the point clouds** to remove noise and enhance structural clarity.
- **Detecting notches** using spatial segmentation techniques, ensuring key feature correspondences.
- **Estimating an initial transformation matrix** via Principal Component Analysis (PCA) to align detected notch regions.
- **Applying stepwise refinement** by analyzing clusters along key axes and performing targeted rotations and translations for improved accuracy.

The experimental results demonstrate the robustness of our method across different viewpoints (Top, Bottom, and Side). The final registration outcomes validate that the real-world point cloud aligns accurately with the reference, particularly around the identified notch regions. The systematic breakdown of alignment steps ensures reproducibility and stability, making it suitable for industrial applications.

Future work could explore adaptive feature extraction techniques or deep learning-based refinement to improve resilience against occlusion and noise. The proposed method offers a reliable and interpretable solution for point cloud alignment.